10-05-00

A

Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

JC849 U.S. PTO 10/03/00

JC926 U.S. PTO 09/678145 10/03/00

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. 002114.P012 Total Pages 2

First Named Inventor or Application Identifier Stuart J. Macdonald, et al.

Express Mail Label No. EL627466591US

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, D. C. 20231

09678145 100300

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. X Specification (Total Pages 39)
(preferred arrangement set forth below)
- Descriptive Title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claims
- Abstract of the Disclosure

3. X Drawings(s) (35 USC 113) (Total Sheets 9)

4. X Oath or Declaration (Total Pages 6)
   a. X Newly Executed (Original or Copy)
   b. ___ Copy from a Prior Application (37 CFR 1.63(d))
   (for Continuation/Divisional with Box 17 completed) **(Note Box 5 below)**
   i. ___ DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. _ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. _ Microfiche Computer Program (Appendix)

7. __ Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
   a. ____ Computer Readable Copy
   b. ____ Paper Copy (identical to computer copy)
   c. ____ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. X Assignment Papers (cover sheet & documents(s))
9. ____ a. 37 CFR 3.73(b) Statement (where there is an assignee)
   X b. Power of Attorney
10. ____ English Translation Document (if applicable)
11. ____ a. Information Disclosure Statement (IDS)/PTO-1449
   ____ b. Copies of IDS Citations
12. ____ Preliminary Amendment
13. X Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. ____ a. Small Entity Statement(s)
   b. Statement filed in prior application, Status still proper and desired
15. ____ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. X Other: Certificate of Express Mail with copy of postcard showing contents of Express Mail package.

17. **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:

___ Continuation ___ Divisional ___ Continuation-in-part (CIP)

of prior application No: _

18. **Correspondence Address**

X Customer Number or Bar Code Label 008791
(Insert Customer No. or Attach Bar Code Label here)

or

X Correspondence Address Below

NAME Sheryl Sue Holloway, Esq. Reg. No. 37,850
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

ADDRESS 12400 Wilshire Boulevard
Seventh Floor

CITY Los Angeles STATE California ZIP CODE 90025-1026

Country U.S.A. TELEPHONE (408) 720-8598 FAX (408) 720-9397

PTO/SB/17(08-00)
Approved for use through 10/31/2002. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL FOR FY 2001

TOTAL AMOUNT OF PAYMENT ($) $1250.00

Complete if Known:
Application No. Unassigned
Filing Date Unassigned
First Named Inventor ____
Group Art Unit Unassigned
Examiner Name Unassigned
Attorney Docket No. 002114.P012

METHOD OF PAYMENT (check one)

1. [X] **The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:**

Deposit Account Number 02-2666
Deposit Account Name ____

[X] **Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17**

2. X **Payment Enclosed:**
X **Check**
____ **Money Order**
____ **Other**

FEE CALCULATION

1. **BASIC FILING FEE**

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	710	201	355	Utility application filing fee	710.00
106	320	206	160	Design application filing fee	
107	490	207	245	Plant filing fee	
108	710	208	355	Reissue filing fee	
114	150	214	75	Provisional application filing fee	

SUBTOTAL (1) $ 710.00

2. **EXTRA CLAIM FEES**

			Extra Claims		Fee from below		Fee Paid
Total Claims	30	− 20** =	10	X	18	=	180.00
Independent Claims	7	− 3** =	4	X	80	=	320.00
Multiple Dependent						=	

****Or number previously paid, if greater; For Reissues, see below.**

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	80	202	40	Independent claims in excess of 3
104	270	204	135	Multiple dependent claim, if not paid
109	80	209	40	**Reissue independent claims over original patent
110	18	210	9	**Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) $ 500.00

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for response within first month	
116	390	216	195	Extension for response within second month	
117	890	217	445	Extension for response within third month	
118	1,390	218	695	Extension for response within fourth month	
128	1,890	228	945	Extension for response within fifth month	
119	310	219	155	Notice of Appeal	
120	310	220	155	Filing a brief in support of an appeal	
121	270	221	135	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive unavoidably abandoned application	
141	1,240	241	620	Petition to revive unintentionally abandoned application	
142	1,240	242	620	Utility issue fee (or reissue)	
143	440	243	220	Design issue fee	
144	600	244	300	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	$40.00
146	710	246	355	For filing a submission after final rejection (see 37 CFR 1.129(a))	
149	710	249	355	For each additional invention to be examined (see 37 CFR 1.129(b))	
179	710	279	355	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify) ______________________ ________

Other fee (specify) ______________________ ________

SUBTOTAL (3) $ 40.00

*Reduced by Basic Filing Fee Paid

SUBMITTED BY:

Typed or Printed Name: Sheryl Sue Holloway

Signature: [illegible] Date: October 3, 2000

Reg. Number: 37,850 Telephone Number: 408-720-8300

002114.P012 PATENT

UNITED STATES PATENT APPLICATION

for

MULTI-LAYER PROTOCOL REASSEMBLY THAT OPERATES INDEPENDENTLY OF UNDERLYING PROTOCOLS, AND RESULTING VECTOR LIST CORRESPONDING THERETO

Applicants:

Stuart J. Macdonald
Jerome N. Freedman

prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard
Los Angeles, CA 90026-1026
(408) 720-8598

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL627466591US
Date of Deposit October 3, 2000
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

MICHELLE BEGAY
(Typed or printed name of person mailing paper or fee)

[illegible]
(Signature of person mailing paper or fee)

MULTI-LAYER PROTOCOL REASSEMBLY THAT OPERATES INDEPENDENTLY OF UNDERLYING PROTOCOLS, AND RESULTING VECTOR LIST CORRESPONDING THERETO

FIELD OF THE INVENTION

This invention relates generally to computer networks, and more particularly to reassembling protocol data flows within a computer network.

COPYRIGHT NOTICE/PERMISSION

BACKGROUND OF THE INVENTION

Communication links between two computers on a network, such as the Internet or a local-area network, are subject to various types of degradation and failure conditions. Protocol analysis is frequently used to determine where potential problems exist in a network. Each network protocol requires the development of a protocol interpreter designed around the characteristics of a particular protocol. Because a network may implement one or more of the over 430 communication protocols currently in common use, a general purpose protocol analysis system must incorporate many individual protocol interpreters.

Although the characteristics of each protocol are different, certain operations in performing protocol analysis are common, such as parsing a protocol data unit to extract a payload. Having a generalized base model for the common operations would save development time in creating the protocol interpreters and reduce the complexity of a general purpose protocol analysis system.

SUMMARY OF THE INVENTION

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

A segmentation and re-assembly (SAR) decode engine reassembles messages from protocol data units exchanged in a communications channel between two computers. The SAR decode engine creates a protocol flow object to represent each protocol layer used in the communications channel. Each of the protocol flow objects has a primary and an alternate circuit element to which are linked circuit flow objects representing protocol data units for the next higher protocol, and the circuit flow objects are linked to the circuit element corresponding to the transmission direction in the channel of the protocol data units represented by the circuit flow objects. The SAR decode engine logically arranges the protocol flow objects in a tree structure corresponding to a hierarchical arrangement of the protocol layers used in the channel. The SAR decode engine logically links the circuit flow objects in a sequence when specified by the associated protocol. The messages in the channel are reassembled from the circuit flow objects linked to the protocol flow object that represents the top layer protocol. The SAR decode engine stores the protocol and circuit flow objects in a database. In one aspect, vector lists are used for circuit flow objects to represent protocol data units that are the result of fragmenting a protocol data unit from a higher layer protocol.

The SAR decode engine of the present invention provides generalized parsing and decoding functions that were previously required to be individually coded in each protocol interpreter. The SAR decode engine also manages data flow storage structures that are common for all protocol interfaces, further reducing the complexity of the individual protocol interpreter and eliminating the need for specialized interfaces previously required to pass data from layer to layer. Because the common functions and storage structures are centralized in the SAR decode engine, the operations can be optimized to improve the overall performance of a protocol analysis system that incorporates the present invention.

The present invention describes systems, clients, servers, methods, and computer-readable media of varying scope. In addition to the aspects and advantages of the present invention described in this summary, further aspects and advantages of the invention will become apparent by reference to the drawings and by reading the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of one embodiment of an operating environment suitable for practicing the present invention;

FIG. 2 is a diagram of one embodiment of a computer system suitable for use in the operating environment of FIG. 1;

FIG. 3 is a diagram illustrating a system-level overview of an embodiment of the invention;

FIG. 4A is a diagram of a protocol flow object data structure for use in an embodiment of the invention;

FIG. 4B is a diagram of flow tree data structure for use in an embodiment of the invention;

FIG. 5A is a diagram illustrating circuit flow objects created by an embodiment of the invention;

FIG. 5B is a diagram illustrating an embodiment of a vector for a circuit flow object in FIG. 5A;

FIGs. 6A-B are flowchart of methods to be performed by a computer according to an embodiment of the invention; and

FIG. 7 is a diagram illustrating an example of a flow tree created by the method of FIG. 6A.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of embodiments of the invention, reference is made to the accompanying drawings in which like references indicate similar elements, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical, functional and other changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

The detailed description is divided into four sections and a conclusion. In the first section, the hardware and the operating environment in conjunction with which embodiments of the invention may be practiced are described. In the second section, a system level overview of the invention is presented. In the third section, methods for an embodiment of the invention are provided. In the fourth section, a particular implementation of the invention is described.

Operating Environment

The following description of FIGs. 1 and 2 is intended to provide an overview of computer hardware and other operating components suitable for implementing the invention, but is not intended to limit the applicable environments. One of skill in the art will immediately appreciate that the invention can be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network.

FIG. 1 shows several computer systems that are coupled together through a network 103, such as the Internet. The term "Internet" as used herein refers to a network of networks which uses certain protocols, such as the TCP/IP protocol, and possibly other protocols such as the hypertext transfer protocol (HTTP) for hypertext markup language (HTML) documents that make up the World Wide Web (web). The physical connections of the Internet and the protocols and communication procedures of the Internet are well known to those of skill in the art. Access to the Internet 103 is typically provided by Internet service providers (ISP), such as the ISPs 105 and 107. Users on client systems, such as client computer systems 121, 125, 135, and 137 obtain access to the Internet through the Internet service providers, such as ISPs 105 and 107. Access to the Internet allows users of the client computer systems to exchange information, receive and send e-mails, and view documents, such as documents which have been prepared in the HTML format. These documents are often provided by web servers, such as web server 109 which is considered to be "on" the Internet. Often these web servers are provided by the ISPs, such as ISP 105, although a computer system can be set up and connected to the Internet without that system being also an ISP as is well known in the art.

The web server 109 is typically at least one computer system which operates as a server computer system and is configured to operate with the protocols of the World Wide Web and is coupled to the Internet. Optionally, the web server 109 can be part of an ISP which provides access to the Internet for client systems. The web server 109 is shown coupled to the server computer system 111 which itself is coupled to web content 110, which can be considered a form of a media database. It will be appreciated that while two computer systems 109 and 111 are shown in FIG. 1, the web server system 109 and the server computer system 111 can be one computer system having different software components providing the web server functionality and the server functionality provided by the server computer system 111 which will be described further below.

Client computer systems 121, 125, 135, and 137 can each, with the appropriate web browsing software, view HTML pages provided by the web server 109. The ISP 105 provides Internet connectivity to the client computer system 121 through the modem interface 123 which can be considered part of the client computer system 121. The client computer system can be a personal computer system, a network computer, a Web TV system, or other such computer system. Similarly, the ISP 107 provides Internet connectivity for client systems 125, 135, and 137, although as shown in FIG. 1, the connections are not the same for these three computer systems. Client computer system 125 is coupled through a modem interface 127 while client computer systems 135 and 137 are part of a LAN. While FIG. 1 shows the interfaces 123 and 127 as generically as a "modem," it will be appreciated that each of these interfaces can be an analog modem, ISDN modem, cable modem, DSL (digital subscriber line) router, satellite transmission interface (e.g. "Direct PC"), or other interfaces for coupling a computer system to other computer systems. Client computer systems 135 and 137 are coupled to a LAN 133 through network interfaces 139 and 141, which can be Ethernet network or other network interfaces. The LAN 133 is also coupled to a gateway computer system 131 which can provide firewall and other Internet related services for the local area network. This gateway computer system 131 is coupled to the ISP 107 to provide Internet connectivity to the client computer systems 135 and 137. The gateway computer system 131 can be a conventional server computer system. Also, the web server system 109 can be a conventional server computer system.

Alternatively, as is well-known, a server computer system 143 can be directly coupled to the LAN 133 through a network interface 145 to provide files 147 and other services to the clients 135, 137, without the need to connect to the Internet through the gateway system 131.

FIG. 2 shows one example of a conventional computer system that can be used as a client computer system or a server computer system or as a web server system. It will also be appreciated that such a computer system can be used to perform many of the functions of an Internet service provider, such as ISP 105. The computer system 201 interfaces to external systems through the modem or network interface 203. It will be appreciated that the modem or network interface 203 can be considered to be part of the computer system 201. This interface 203 can be an analog modem, ISDN modem, cable modem, DSL router, token ring interface, satellite transmission interface (e.g. "Direct PC"), or other interfaces for coupling a computer system to other computer systems. The computer system 201 includes a processor

205, which can be a conventional microprocessor such as an Intel Pentium microprocessor or Motorola Power PC microprocessor. Memory 209 is coupled to the processor 205 by a bus 207. Memory 209 can be dynamic random access memory (DRAM) and can also include static RAM (SRAM). The bus 207 couples the processor 205 to the memory 209 and also to non-volatile storage 215 and to display controller 211 and to the input/output (I/O) controller 217. The display controller 211 controls in the conventional manner a display on a display device 213 which can be a cathode ray tube (CRT) or liquid crystal display. The input/output devices 219 can include a keyboard, disk drives, printers, a scanner, and other input and output devices, including a mouse or other pointing device. The display controller 211 and the I/O controller 217 can be implemented with conventional well known technology. A digital image input device 221 can be a digital camera which is coupled to an I/O controller 217 in order to allow images from the digital camera to be input into the computer system 201. The non-volatile storage 215 is often a magnetic hard disk, an optical disk, or another form of storage for large amounts of data. Some of this data is often written, by a direct memory access process, into memory 209 during execution of software in the computer system 201. One of skill in the art will immediately recognize that the term "computer-readable medium" includes any type of storage device that is accessible by the processor 205 and also encompasses a carrier wave that encodes a data signal.

It will be appreciated that the computer system 201 is one example of many possible computer systems which have different architectures. For example, personal computers based on an Intel microprocessor often have multiple buses, one of which can be an input/output

(I/O) bus for the peripherals and one that directly connects the processor 205 and the memory 209 (often referred to as a memory bus). The buses are connected together through bridge components that perform any necessary translation due to differing bus protocols.

Network computers are another type of computer system that can be used with the present invention. Network computers do not usually include a hard disk or other mass storage, and the executable programs are loaded from a network connection into the memory 209 for execution by the processor 205. A Web TV system, which is known in the art, is also considered to be a computer system according to the present invention, but it may lack some of the features shown in FIG. 2, such as certain input or output devices. A typical computer system will usually include at least a processor, memory, and a bus coupling the memory to the processor.

It will also be appreciated that the computer system 201 is controlled by operating system software which includes a file management system, such as a disk operating system, which is part of the operating system software. One example of an operating system software with its associated file management system software is the operating system known as Windows '95® from Microsoft Corporation of Redmond, Washington, and its associated file management system. The file management system is typically stored in the non-volatile storage 215 and causes the processor 205 to execute the various acts required by the operating system to input and output data and to store data in memory, including storing files on the non-volatile storage 215.

System Level Overview

A system level overview of the operation of an embodiment of a segmentation and re-assembly (SAR) decode engine according to the invention is described by reference to FIGs. 3, 4A, 4B, 5A and 5B. Beginning with FIG. 3, a communication channel 320 is established between two computers, computer A 301 and computer B 303. Computer B 303 can be a client, such as client computer systems 121, 125, 135, 137 in FIG. 2, connected through the Internet 103 or LAN 133 (the communications channel 320) to computer A 301 functioning as a server such as server computer systems 111 or 143. As is conventional, the data flowing through the communication channel 320 is encoded into "protocol data units" according to a multi-layered data communication protocol, such as defined in the OSI (Open Systems Interconnection) model. Frequently, protocol data units exchanged at the lowest protocol layer are referred to as "frames," while those at the higher protocol layers are referred to as "packets." For simplicity in describing the invention, the data exchanged at all layers is referred to herein as protocol data units or PDUs, and such usage is further clarified with the number or name of the corresponding protocol layer when appropriate.

Protocol data units in the communications channel 320 are captured in a frame capture buffer 305 and retrieved by the SAR decode engine 307. Multiple protocol interpreters, collectively shown at 311, are used by the SAR decode engine 307 to determine the appropriate sequencing or reassembly of the data into the data flow recognized by a particular protocol layer. The SAR decode engine 307 creates various flow objects to represent the data flows at each level and stores the flow objects in a flow object database 309 as described next.

The SAR decode engine 307 is also responsible for unpacking the PDUs in creating the flow objects, thus eliminating the need for each of the protocol interpreters 311 to contain code that does the repetitive unpacking operations.

The SAR decode engine 307 creates protocol flow objects to represent the protocol layers in the communication channel 320 and circuit flow objects to represent the data as it is decoded by the protocols at each level. One embodiment of a protocol flow object data structure is shown in FIG. 4A. The protocol flow object 400 contains a key 401 used to identify the particular protocol flow object within the flow object database 309. The protocol flow object 401 also contains two circuit elements that link the circuit flow objects to the protocol flow object 401. A primary circuit element 403 is linked to a series of circuit flow objects that represent the data being transmitted in one direction between the computers 301 and 303 and define a one-way circuit 321 in the communications channel 320. An alternate circuit element 405 is linked to a series of circuit flow objects that define the opposite circuit 323 within the channel 320. In the present embodiment, the primary circuit is determined by the transmission direction of the first protocol data unit that is received in the frame capture buffer but it will be appreciated that the primary and alternate circuits can be pre-determined based on various criteria, such as the whether the source computer functions as the client or server in a client-server network. It will further be appreciated that the key and the logical links can be address pointers, hash table values, or similar data structures conventionally used to locate and relate records within a data base or other data organization. For example, in the implementation discussed further below, a hash table is used.

The protocol flow objects created for the channel 320 are logically linked together by the SAR decode engine 470 in a hierarchical flow tree data structure. Using an Ethernet network and the standard TCP/IP protocol stack as an example, a corresponding flow tree 420 shown in FIG. 4B has at its base a root flow object 421, which is linked to a data link layer protocol flow object, shown as DLC protocol object 423. The network layer protocol is the Internet Protocol (IP) and is represented in the tree 420 by the IP protocol flow object 425. In the present example, there are two connections between the computers at the transport protocol layer, one for retrieving HTML formatted web pages using the HTTP application protocol and one for retrieving data from a Microsoft SQL database using a tabular data stream (TDS) protocol. Therefore, two TCP protocol flow objects are created at the transport layer and linked to the IP protocol flow object 425 in the tree 420, one for each connection. TCP protocol flow object 427 represents the connection between the two computers used to transport the requests for web pages and the corresponding web pages, while TCP protocol flow object 429 represents the connection that transports the SQL commands and resulting data. Similarly, there are two protocol flow objects at the application protocol level of the tree 420, an HTTP protocol flow object 431 and a MS SQL protocol flow object 433, linked to their respective TCP protocol objects.

The key 401 for each protocol object may be either a source identifier when it alone is sufficient to specify the appropriate protocol object, or a combination of both source and destination identifiers. One of in the art will immediately recognize that the tree 420 shown in FIG. 4B is a simplified version of the types of hierarchical flow trees that can be created for the connections between two computers.

Although not illustrated in FIG. 4B, each of the protocol flow objects in the tree 420 is further linked to the circuit flow objects that represent the primary and alternate circuits of the connection at that level. The circuit flow objects linked to a protocol flow object for a particular protocol layer represent the payloads of the protocol data units for that layer. The configuration of the circuit flow object depends upon the characteristics of the associated protocol layer. FIG. 3 is used in conjunction with FIGs. 5A and 5B to describe examples of the circuit flow objects created by the SAR decode engine 307 for each protocol layer in a simplified three-layer protocol stack corresponding to protocol interpreter A 313, protocol interpreter B 315, and protocol interpreter C 317 in FIG. 3.

At the source computer, the top layer protocol A receives a message 501 from an application. Protocol A appends a header 505 to the message 501 to create a protocol A PDU 503. The protocol A PDU 503 is then fragmented by the middle layer protocol B into three protocol B PDUs 513, 515, 517. Protocol B PDU 513 contains a header 519, with the first portion of the message 507 as its payload 521. Protocol B PDU 515 contains a header 523 and a second portion of the message 509 as its payload 525. Similarly, the final protocol B PDU 517 contains a header 527 and the final portion of the message 511 as its payload 529. The protocol B PDUs 513, 515, 517 are transmitted over the communication channel 320 by the bottom layer protocol C as protocol C PDUs. For ease in illustration, the protocol C PDUs are not shown and assumed to have a one-to-one correspondence to the protocol B PDUs 513, 515, 517.

The SAR decode engine 307 retrieves the first protocol C PDU, i.e., having the protocol B PDU 513 in its payload, from the frame capture buffer 305 and determines the lowest level protocol is protocol C. The SAR decode engine 307 creates a root protocol flow object and protocol flow object for protocol C if they do not already exist. The SAR decode engine calls protocol interpreter C 317 and creates a circuit flow object 531 corresponding to protocol B PDU 513 from the payload of the protocol C PDU. The remaining protocol C PDUs are retrieved from the frame capture buffer 305 and passed to the protocol interpreter C 317 one at a time. As instructed by the protocol interpreter C 317 , the SAR decode engine creates circuit flow objects 533, 537 from the payloads extracted from the remaining protocol C PDUs. In one embodiment, an extracted PDU is stored as a record within the flow object database 309 along with an identifier, while in another embodiment the extracted PDU is held in a data buffer and accessed by its address within the data buffer. Other alternate embodiments of the circuit flow objects for a non-fragmented protocol will be readily apparent to one of skill in the art and are considered within the scope of the invention.

The protocol interpreter C 317 also specifies a sequence order for the protocol C PDUs and the SAR decode engine links the circuit flow objects 531, 533, 537 in the specified order as shown by single arrows 587 and 539. Thus, the linked circuit flow objects 531, 533, 537 represent a linked list of protocol B PDUs 513, 515, 517 that form one of the circuits 321 or

323 in the communications channel 320. The protocol interpreter C 317 also informs the SAR decode engine that the middle layer protocol is protocol B.

Because the original protocol A PDU 503 was fragmented by protocol B, the circuit flow object linked to the protocol flow object for protocol B is a vector list containing vectors 541, 543, 545 that locate the fragments of the PDU 503 within the circuit flow objects 531, 533, 535. The vectors 541, 543, 545 are formatted as shown in FIG. 5B. Each vector consists of the number 551 of the corresponding protocol B PDU, a length 553 of the fragment contained in the protocol B PDU (in bytes), and an offset 555 for the beginning of the fragment within the protocol B PDU. The information for the vectors is obtained by the SAR decode engine 307 by calling the protocol interpreter B 515 and passing in the circuit flow objects 531, 533, 535 in sequence order. The vector list is then linked to the protocol flow object for protocol B. Protocol interpreter B 515 designates protocol A as the next protocol layer.

Now, the SAR decode engine 307 can reassemble the original message 501. The SAR decode engine 307 extracts the data from the circuit flow objects 531, 533, 535 as specified by the vectors 541, 543, 545 into a re-assembly buffer 547. The SAR decode engine 307 calls the protocol interpreter A 313, passing in the re-assembly buffer 547. The protocol interpreter A 313 returns instructions to the SAR decode engine 307 on how extract the message 501 from the re-assembly buffer 547. A flow object 549 containing the message 501 is linked to the protocol flow object for protocol A.

The system level overview of the operation of an embodiment of the invention has been described in this section of the detailed description. A segmentation and reassembly (SAR) decode engine receives protocol data units from a communication channel between two computers, sequences the protocol data units, and re-assembles the data in the protocol data units into the messages exchanged by the computers. The SAR is responsible for unpacking the payloads from the protocol data units as instructed by a protocol interpreter associated with the protocol layer that created the protocol data units, and for creating and maintaining a flow object database that holds flow objects representing the data flows at each protocol layer. The flow objects are arranged in a hierarchical flow tree data structure corresponding to the layers in the protocol stack. The flow objects at the top of the tree are used to re-assemble the messages. While the invention is not limited to any particular configuration of data structures, sample embodiments of flow objects and flow trees have been described. For example, one of skill in the art will readily appreciate that the circuit flow objects may contain pointers to the corresponding lowest layer protocol data units captured within the frame buffer and offset information for parsing the protocol data units at the various protocol layers instead of containing the actual payloads of the protocol data units as extracted by the SAR.

Methods of Embodiments of the Invention

In the previous section, a system level overview of the operations of embodiments of the invention was described. In this section, the particular methods of the invention are described in terms of computer software with reference to a series of flowcharts. The methods to be performed by a computer constitute computer programs made up of computer-executable instructions. Describing the methods by reference to a flowchart enables one skilled in the art to develop such programs including such instructions to carry out the methods on suitably configured computers (the processor of the computer executing the instructions from computer-readable media). The computer-executable instructions may be written in a computer programming language or may be embodied in firmware logic. If written in a programming language conforming to a recognized standard, such instructions can be executed on a variety of hardware platforms and for interface to a variety of operating systems. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein. Furthermore, it is common in the art to speak of software, in one form or another (e.g., program, procedure, process, application, module, logic...), as taking an action or causing a result. Such expressions are merely a shorthand way of saying that execution of the software by a computer causes the processor of the computer to perform an action or a produce a result.

Turning now to FIG. 6A, the acts to be performed by a computer executing one embodiment of an SAR method 600 are shown and described with further reference to FIG. 7 that illustrates a protocol tree created by the SAR method 600. The root flow object has been omitted from FIG. 7 for ease in illustration. In general, each protocol interpreter called by the SAR method 600 returns instructions that direct the SAR method 600 in extracting the payload from each circuit flow object created by the immediate lower layer protocol.

Additionally, the protocol interpreter for a protocol N that fragments the protocol data units from the higher layer N+1 specifies the position of the fragments within the N+1 PDU. Protocol interpreters may also specify the sequence in which the circuit flow objects must be processed by the next higher layer protocol. Although not shown or described in this section, one of skill in the art will immediately recognize that various error recovery functions can be incorporated into the SAR method 600 to handle transmission problems at the lowest protocol level, such as out-of-sequence frames, duplicated/retransmitted frames, missing frames and the like.

The protocol tree in FIG. 7 assumes an Ethernet network running TCP/IP and an HTTP connection between the two computers. The HTTP protocol layer fragments data that is greater than a pre-defined length and creates multiple HTTP PDUs, each having a fragment as its payload, to hold the data. The HTTP protocol designates each PDU as a first, last, or middle PDU when the data is spread over multiple PDUs, i.e., multi-PDU data, or as a single PDU if the data is unfragmented, single-PDU data. For simplicity in explanation, it is assumed that neither the TCP nor IP protocol layers fragment PDUs from a higher layer, but that the TCP protocol layer does sequence the PDUs it receives from the HTTP layer. The protocol flow objects in FIG. 7 are keyed as shown in Table 1.

Layer	Protocol	Key	Example
Data Link	Ethernet (DLC)	Source & destination NIC addresses (key 703)	Computer A: D5C3FF (6 bytes) Computer B: 29D0A6
Network	IP	Source & destination IP addresses (key 727)	A: 161.69.10.165 (4 bytes) B: 161.69.10.164
Transport	TCP	Source & destination port	A: 80 (2 bytes)

Layer	Protocol	Key	Example
		addresses (key 751)	B: 1908
Application	HTTP	Source port (key 775)	A: 1908 (2 bytes)

Table 1. Example Protocol Layers and Keys

The SAR method 600 retrieves the first Ethernet PDU from a capture memory, a trace file, or similar frame buffering facility (block 601) and examines the Ethernet PDU header to determine the protocol used at the data link layer (block 603). In an alternate embodiment, the SAR method 600 obtains a range of Ethernet PDU numbers that are to be processed and retrieves each Ethernet PDU in turn by number. The SAR method 600 creates a root flow object and a protocol flow object for the data link layer protocol (block 605), shown in FIG. 7 as DLC protocol flow object 701. The SAR method 600 calls the DLC protocol interpreter specific to the Ethernet protocol with the first Ethernet PDU (block 607). The payload from the first Ethernet PDU is extracted according to the instructions returned from the DLC protocol interpreter and used to create a first circuit flow object 709 (block 609). The circuit flow object 709 is linked to the primary circuit element 705 (block 611) since in this embodiment, the first PDU received defines the primary circuit. If more Ethernet PDUs remain in the frame buffer (block 613), each is retrieved and passed into the protocol interpreter at block 607 and the cycle repeats until the SAR has created a circuit flow object from each Ethernet PDU in the frame buffer, e.g., circuit flow objects 709, 711, 713, 715, 717, 719, 721, 723, and linked the circuit flow objects into the appropriate circuit element 705, 707. If the current protocol interpreter has not specified the protocol for the next layer (block 617), the SAR method 600 exits.

Assuming that the DLC protocol interpreter has designated IP as the network layer, the SAR method 600 creates an IP protocol flow object 725 (block 619), retrieves the first circuit flow object 709 (block 621), and calls the IP protocol interpreter (block 623), passing in the circuit flow object 709. In the present example, the IP protocol does not fragment TCP PDUs (block 625), so the SAR method 600 creates a circuit flow object 733 from the payload in the circuit flow object 709 (block 629). The SAR method 600 links the circuit flow object 733 to the primary circuit element 729 in the IP protocol flow object (block 631). The creation and linking process is repeated for each circuit flow object linked to the DLC protocol flow object 725 (blocks 623 and 635), resulting in circuit flow objects 733, 735, 737, 739, 741, 743, 745, 747 that correspond to the payloads of circuit flow objects 709, 711, 713, 715, 717, 719, 721, 723. The circuit flow objects 733, 735, 737, 739, 741, 743, 745, 747 are linked into the circuit elements 729, 731 in the IP protocol flow object 725 in an order corresponding to the order of the circuit flow objects 709, 711, 713, 715, 717, 719, 721, 723. The IP protocol interpreter also specifies TCP as the protocol for the next higher layer, i.e., the transport layer, so the test at block 617 returns control to block 619 to process the TCP protocol layer.

The SAR method 600 creates a TCP protocol flow object 749 at block 691, retrieves the first circuit flow object 733 at block 621, and calls the TCP protocol interpreter at block 523. The SAR method creates a circuit flow object 757 from the payload in the circuit flow object 733 at block 629 and links the circuit flow object 757 to the primary circuit element 753 in the TCP protocol flow object 749 at block 631. Remembering that the TCP protocol layer sequences HTTP PDUs (block 637), when all circuit flow objects at the TCP level have been created from the corresponding circuit flow objects at the IP level and linked to the appropriate circuit element, the SAR method creates sequence links 1, 2, 3, 4, 5, 6, 7 and 8 to establish the proper sequencing of the circuit flow objects for the next higher layer, i.e., the application layer, at block 639. The TCP protocol interpreter specifies HTTP as the application protocol layer so when the sequencing is complete at bock 639, the test at block 617 returns control to block 619 to process the HTTP protocol layer.

Once the SAR method 600 has created the HTTP protocol flow object 773 at block 619, its passes each circuit flow object 757, 759, 761, 763, 765, 769, 771, 775 to the HTTP protocol interpreter in the order specified by the sequence links so that the circuit flow object 757 is processed first, the circuit flow object 765 is processed second, and so on, with the circuit flow object 771 being processed last. Because the HTTP protocol fragments data, in addition to instructing the SAR method on how to extract the data from the circuit flow objects linked to the TCP protocol flow object 749, at block 623 the HTTP protocol interpreter also returns a position designation (first, middle, last, single) for each of the circuit flow objects. The SAR method 600 creates vector lists 781, 783, 785, 791, 797, 799 as circuit flow objects for the HTTP layer (block 627) as follows. For the circuit flow objects 757, 765 and 759, the SAR method 600 creates vector lists 781, 783 and 785, each containing a single vector representing an HTTP PDU that contains unfragmented data. When the SAR method 600 passes the circuit flow object 767 to the HTTP protocol interpreter, the protocol interpreter informs it that circuit flow object 767 is the first PDU of multi-PDU data, so the SAR method creates a vector 787 associated with the HTTP PDU represented by the circuit flow object 767. Next, when the SAR method 600 passes the circuit flow object 769 to the HTTP protocol interpreter, the protocol interpreter informs it that the circuit flow object 769 is the final PDU for the multi-PDU data and the SAR method 600 creates a vector 789 associated with the HTTP PDU represented by the circuit flow object 769. Since all the data fragments have now been received, the SAR method 600 creates the vector list 791 from the vectors 787 and 789 to represent the multi-PDU data. Similarly, the processing of circuit flow objects 761 and 763 result in a vector list 797 containing two vectors 793 and 795. The final circuit flow object 771 is used to create a vector list 799 for the corresponding single-PDU data. Because HTTP is the top protocol layer, once all the vector lists have been created, the test at block 617 is false and the SAR method 600 exits.

Because vector lists, such as vector lists 781, 783, 785, 791, 797 and 799 in FIG. 7, represent fragmented data, a supporting method illustrated in FIG. 6B is used by the SAR engine to reassemble the data from vector lists. The reassembly method 650 extracts the data fragments from the corresponding PDUs as specified by the information in the vectors in a vector list (block 651) and creates the de-fragmented data in a data buffer (block 653). The buffer address and length of the de-fragmented data is returned to the SAR engine (block 655). If the de-fragmented data represents the actual message exchanged between the computers in the communications channel, no further processing by the SAR engine is necessary. Otherwise, the de-fragmented data is treated as a circuit flow object for a protocol layer N and is used to create the circuit flow object for the protocol layer N+1 as described above in conjunction with FIG. 6A.

The particular method performed by computer when operating as the SAR decode engine in one embodiment have been described. A SAR method performed by the computer has been shown by reference to flowcharts in FIGs. 6A-B including all the acts from 601 until 639 and from 651 until 655, with an example flow tree created by the embodiment of the SAR method shown in FIG. 6A having been illustrated in FIG. 7.

SAR Decode Engine Implementation

In this section of the detailed description, a particular implementation of the SAR decode engine is presented in terms of an application program interface (API), a set of size parameters, and an error handling methodology.

SarAddDu() API

The *SarAddDu*() API is used by a protocol interpreter (PI) to instruct the SAR decode engine to extract the payload of a protocol data unit (PDU) to a circuit flow object, to properly sequence the newly added circuit flow object, and to associate that circuit flow object with a given protocol. The arguments for *SarAddDu*() are as follows:

hInterp
A handle to a particular instance of a data structure, PIINTERP used by the SAR decode engine. There is one PIINTERP per instance of SAR. This data consists of generic information set up by each protocol, parse information to enable protocols to complete their tasks, and PDU data.

uOffset
Offset of the data relative to the start of the start of the possibly reassembled data.

uTotalLength
Total Length of PDU(optional). If unused/ unknown then set 0.

uFragLength

PDU/Frag Length per header. Length of data starting at uOffset. Can exceed the size of the current PDU. Example, DCE RPC header claims there are 5800 bytes of data. Set uFragLength to 5800 bytes. This will cause DCE RPC to claim the next 5800 bytes of data from an underlying STREAM such as TCP. This only works on protocols such as TCP which are classified as STREAM.

ulSequence,

Sequence number, if applies. Otherwise, set to 0.

ulID,

ID, if applies. Otherwise, set to 0.

uPosFlags,

SAR_FIRST,	First of multiple fragments in PDU.
SAR_MIDDLE,	Middle of multiple fragments in PDU.
SAR_LAST,	Last of multiple fragments in PDU.
SAR_ONLY,	Unfragmented data in PDU.
SAR_STREAM,	If a protocol is a STREAM.
SAR_HAS_A_HEADER	Set if protocol contains a valid header.

uProtoID

Protocol ID of Protocol Interpreter which will be used to parse the data. For example, IP associates TCP/UDP… with re-assembled data.

The *SarAddDu*() API is called for all PDUs for which there is data. TCP, for example, does not call this function when the TCP segment contains no user data. This is so that in the event there are PDUs for multi-PDU data interspersed with PDUs for single-PDU data, the processing of all PDUs for any given circuit will be in a time-ordered manner. *SarAddDu*() allows PIs to delineate their data based on position (First/ Middle/ Last…) and to associate a "next" protocol with it. For example, if the next protocol is UDP, the IP PI references IP.Data[0] with IP.Hdr.length bytes- IP header size bytes for each middle and last IP PDU and associates it with UDP. In the next pass, the IP Sequences are parsed and the UDP PDUs are re-assembled before the UDP PI is called.

A simple example based on the Microsoft SQL Server TDS protocol illustrates the use of the *SarAddDu*() API. In TDS, there is an 8 byte header which indicates the TDS command type, a status flag which indicates whether or not a given PDU is the last in a message (PDU), as well as a two byte field indicating the length of the PDU. Using two sample PDU, PDUs 148 and 150, there is response to a SQL query, which returns 30 rows of data spanning the two PDUs and consisting of 842 bytes of re-assembled data. Even though PDU 148 has 512 bytes of TDS data (indicated by the TCP layer), the first call to *SarAddDu*() uses SAR_FIRST because the status flag indicates that this PDU is not the last fragment of the TDS message. The second call to *SarAddDu*() for PDU 150, however, uses SAR_LAST, because the status flag indicates that it is the last PDU for the TDS message. In this example, the following call is made to *SarAddDu*():

```
SarAddDU(hInterp,       /* PIINTERP Handle                    */
        uOffset,        /* Start at Data[0]. */
        0,              /* Total Length of PDU.      */
        uLength,        /* PDU/Frag Length per header. */
        0,              /* Sequence number, if applies        */
        0,              /* ID, if applies.                    */
        uSarFlags,      /* FIRST, CONT, LAST                  */
        PROTO_TDS);     /* Protocol ID to associate with Data.  */
```

In this call, *uOffset* is set to the first byte of the TDS header if the PDU is specified as SAR_FIRST, otherwise it is set to the start of the TDS data, even if there is a header on a continuation PDU. *uLength* is set to the total length specified by the TDS header. It should be noted that a request for more bytes in the call to *SarAddDu*() than are in the PDU will cause SAR to attempt to steal the extra bytes needed from subsequent PDUs.

SAR Sizes

The maximum PDU size in this present implementation is limited to 32KBytes and the maximum size of the re-assembly buffer is 32KBytes. There can be (2^32)-1 PDUs. PDU-1L (0xffffffff) is reserved for internal use. The maximum number of vectors displayed in a vector list is thirty-two. If there are more than thirty-two vectors, the first thirty-one vectors plus the last vector will be processed.

Error Handling

The present implementation of the SAR decode engine will recover out-of-sequence and duplicate frames at the data link layer. Out-of sequence frames are re-sequenced. When a duplicate, i.e., retransmitted, frame is detected, the SAR decode engine substitutes the most recent frame in time order for the earlier frame in the sequence. When a frame is missing, the SAR decode engine processes all frames up to the missing frame through all protocol layers. Truncated frames cause the SAR decode engine to terminate with an error message when the truncated frame is detected.

Conclusion

A segmentation and re-assembly (SAR) decode engine has been described. The SAR decode engine receives protocol data units from a communication channel between two computers, sequences the protocol data units, and re-assembles the data into the messages exchanged by the computers. The SAR decode engine is responsible for unpacking the payloads from the protocol data units as instructed by a protocol interpreter associated with the protocol data unit, and for creating and maintaining a flow object database containing flow objects representing the data flows at each protocol layer. Embodiments of the flow object database and the flow objects have been described, along with a software method executed by a computer acting as the SAR decode engine. Additionally, the particular characteristics of one implementation of the SAR decode engine have bee set forth.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention. The terminology used in this application with respect to networks is meant to include all of network environments that use a layer protocol architecture. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.

CLAIMS

What is claimed is:

1. A computerized method for sequencing and reassembling messages from protocol data units exchanged in a communications channel between two computers, the method comprising:

creating a protocol flow object to represent each protocol layer used by the communications channel, each protocol flow object having a circuit element associated with each transmission direction in the channel;

arranging the protocol flow objects in a logical tree structure that mirrors a hierarchy for the protocol layers;

creating circuit flow objects for each protocol layer to represent the protocol data units for the protocol layer immediately higher in the hierarchy;

associating a transmission direction with each circuit flow object;

linking each circuit flow object for a protocol layer to the circuit element of the representative protocol flow object that matches the transmission direction associated with the circuit flow object;

sequencing the circuit flow objects linked to a particular protocol flow object when specified by the protocol layer represented by the particular protocol flow object; and

reassembling the messages from the circuit flow objects linked to the protocol flow object at the top of the tree structure.

2. The method of claim 1, wherein creating the circuit flow objects for each protocol layer comprises:

creating the circuit flow objects for the protocol flow object at the bottom of the tree structure by extracting data from the protocol data units for the protocol layer lowest in the hierarchy; and

creating the circuit flow objects for the remaining protocol flow objects in the tree structure by extracting data from the circuit flow objects linked to the protocol flow object immediately lower in the tree structure.

3. The method of claim 1, wherein a circuit flow object comprises a vector list to represent fragmented data.

4. The method of claim 3, wherein a vector list comprises a vector specifying a protocol data unit number, a length value, and an offset value for each fragment of the fragmented data.

5. The method of claim 4 further comprising:

reassembling the fragmented data in accordance with the vectors in a vector list.

6. The method of claim 1 wherein the protocol flow objects are created in order from the bottom to the top of the hierarchy.

7. The method of claim 6, wherein the circuit flow objects for a current protocol flow object are created before creating the protocol flow object for the protocol layer immediately above the current protocol flow object in the hierarchy.

8. The method of claim 1 wherein arranging the protocol flow objects into a logical tree structure comprises:

creating multiple branches in the tree structure when a plurality of protocol layers are immediately above a current protocol layer in the hierarchy, each branch corresponding to one of the plurality of protocol layers.

9. The method of claim 1 further comprising:

determining the protocol layers in the hierarchy.

10. The method of claim 1 further comprising:

storing the protocol flow objects and the circuit flow objects in a flow object database.

11. A computer-readable medium having computer-executable instructions to a cause a computer to perform a method comprising:

creating a protocol flow object to represent each protocol layer used by a communications channel, each protocol flow object having a circuit element associated with a transmission direction in the channel;

arranging the protocol flow objects in a logical tree structure that mirrors a hierarchy for the protocol layers;

creating circuit flow objects for each protocol layer to represent the protocol data units for the protocol layer immediately higher in the hierarchy;

associating a transmission direction with each circuit flow object;

linking each circuit flow object for a protocol layer to the circuit element of the representative protocol flow object that matches the transmission direction associated with the circuit flow object;

sequencing the circuit flow objects linked to a particular protocol flow object when specified by the protocol layer represented by the particular protocol flow object; and

reassembling the messages from the circuit flow objects linked to the protocol flow object at the top of the tree structure.

12. The computer-readable medium of claim 11 having further computer-executable instructions comprising:

creating the circuit flow object for the protocol flow object at the bottom of the tree structure by extracting data from the protocol data units for the protocol layer lowest in the hierarchy; and

creating the circuit flow objects for the remaining protocol layers by extracting data from the circuit flow objects linked to the protocol flow object immediately lower in the tree structure.

13. The computer-readable medium of claim 11 having further computer-executable instructions comprising:

creating a circuit flow object as a vector list to represent fragmented data.

14. The computer-readable medium of claim 13 having further computer-executable instructions comprising:

creating a vector list from a plurality of vectors, each vector specifying a protocol data unit number, a length value, and an offset value for a fragment of the fragmented data.

15. The computer-readable medium of claim 14 having further computer-executable instructions comprising:

reassembling the fragmented data in accordance with the vectors in a vector list.

16. The computer-readable medium of claim 11 having further computer-executable instructions comprising:

creating multiple branches in the tree structure when a plurality of protocol layers are immediately above a current protocol layer in the hierarchy, each branch corresponding to one of the plurality of protocol layers.

17. The computer-readable medium of claim 11 having further computer-executable instructions comprising:

determining the protocol layers in the hierarchy.

18. The computer-readable medium of claim 11 having further computer-readable instructions comprising:

storing the protocol flow objects and the circuit flow objects in a flow object database.

19. A computer-readable medium having stored thereon an protocol flow object data structure comprising:

a key field containing data representing an identifier for a connection between two computers at a protocol layer;

a primary circuit element containing data representing a link to a series of protocol data units flowing in one direction in the connection identified by the key field; and an alternate circuit element containing data representing a link to a series of protocol data units flowing in an opposite direction in the connection identified by the key field.

20. The computer-readable medium of claim 19, wherein the links comprise hash tables for identifying the series of data units.

21. A computer-readable medium having stored thereon a re-assembly vector comprising:

a protocol data unit field containing data representing a number for a protocol data unit;

a length field containing data representing a length of a data payload in the protocol data unit identified by the protocol data unit field; and

an offset field containing data representing a starting position of the data payload in the protocol data unit identified by the protocol data unit field.

22. A computer-readable medium having stored thereon a flow object data structure comprising:

a plurality of protocol flow objects, each protocol flow object comprising:

a key field containing data representing an identifier for a connection between two computers at a protocol layer;

a primary circuit element containing data representing a link to a series of protocol data units flowing in one direction in the connection identified by the key field; and an alternate circuit element containing data representing a link to a series of protocol data units flowing in an opposite direction in the connection identified by the key field;

a tree structure comprising a plurality of entries, each entry comprising:

a protocol field containing data representing the identifier for one of the plurality of protocol flow objects;

a lower protocol field containing data representing the identifier for the protocol flow object immediately lower in a protocol layer hierarchy relative to the protocol flow object identified by the protocol field; and

a higher protocol field containing data representing the identifier for the protocol flow object immediately higher in the protocol layer hierarchy relative to the protocol flow object identified by the protocol field.

23. The computer-readable medium of claim 22 further comprising:

a plurality of circuit flow objects, each circuit flow object containing data representing one of the protocol data units.

24. A computerized system comprising:

a processor;

a memory coupled to the processor through a bus;

a computer-readable medium coupled to the processor through the bus;

a plurality of protocol interpreters stored on the computer-readable medium for execution by the processor; and

a decode engine executed from the computer-readable medium to cause the processor to create protocol flow objects representing protocol layers and circuit flow objects representing data flows at the protocol layers,

extract data from the circuit flow objects representing protocol data units at a particular protocol layer as directed by one of the protocol interpreters,

sequence the circuit flow objects representing the protocol data units at a particular protocol layer if directed by one of the protocol interpreters, and

reassemble messages from the circuit flow objects representing the protocol data units at a particular protocol layer if directed by one of the protocol interpreters.

25. The computer system of claim 24, wherein the decode engine further causes the processor to store the protocol flow objects and circuit flow objects in a flow database, logically link the protocol flow objects into a hierarchical tree structure, and to logically link the circuit flow objects to the protocol flow objects.

26. The computer system of claim 24, wherein the decode engine further causes the processor to create a circuit flow object as a vector list to represent fragmented data.

27. The computer system of claim 26, wherein the decode engine further causes the processor to create a vector list from a plurality of vectors, each vector specifying a protocol data unit number, a length value, and an offset value for a fragment of the fragmented data.

28. The computer system of claim 27, wherein the decode engine further causes the processor to reassemble the fragmented data in accordance with the vectors in a vector list.

29. A method of communicating between a protocol interpreter and a segmentation and re-assembly decode engine for a communications network comprising:

issuing, by the protocol interpreter, an add data unit command;

receiving, by the segmentation and re-assembly decode engine, the add data unit command; and

issuing, by the segmentation and re-assembly decode engine in response to receiving the add data unit command, an instruction to a flow object data base.

30. The method of claim 29, wherein the instruction is selected from the group consisting of adding a circuit flow object to the flow object data base, associating a circuit flow object to a protocol flow object in the flow object data base, retrieving a circuit flow object from the flow object data base, and sequencing a circuit flow object relative to other circuit flow objects in the data base.

ABSTRACT OF THE DISCLOSURE

A segmentation and re-assembly (SAR) decode engine receives protocol data units of data from a communication channel between two computers, sequences the protocol data units, and re-assembles the data in the protocol data units into the messages exchanged by the computers. The SAR decode engine is responsible for unpacking the payloads from the protocol data units as instructed by a protocol interpreter associated with the protocol data unit, and for creating and maintaining a flow object database containing flow objects representing the data flows at each protocol layer. The SAR decode engine creates a protocol flow object for each protocol layer and logically links the protocol flow object to circuit flow objects that define two one-way circuits within the channel. The circuit flow objects linked to a protocol flow object are logical representations of the protocol data units for the next higher protocol layer. For protocols that fragment data, each circuit flow object is a vector list containing one or more vectors that define the length, starting location and position of the data fragments in the immediately lower layer circuit flow objects.

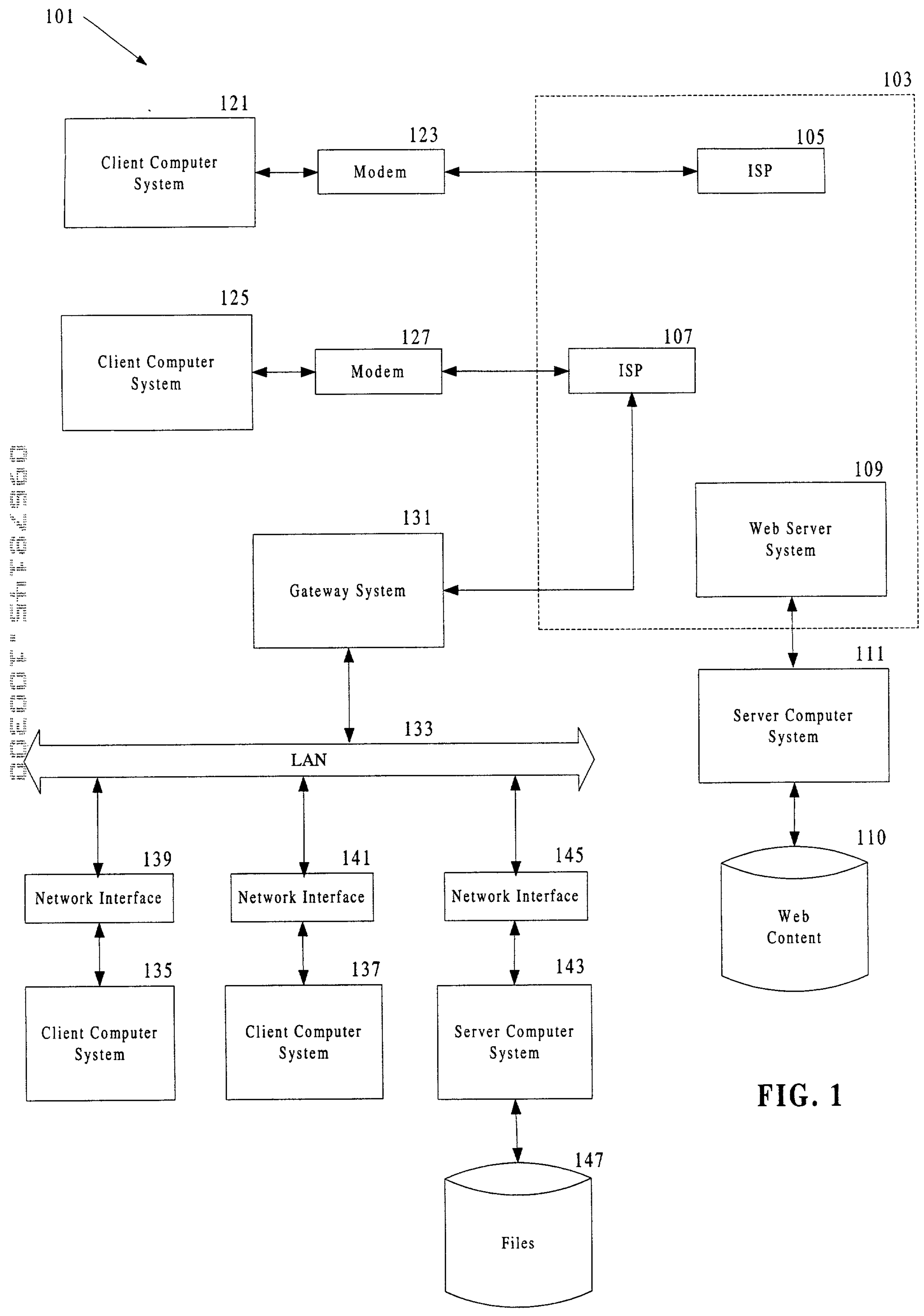

101
103
121
Client Computer System
123
Modem
105
ISP
125
Client Computer System
127
Modem
107
ISP
109
Web Server System
131
Gateway System
111
Server Computer System
133
LAN
110
Web Content
139
Network Interface
141
Network Interface
145
Network Interface
135
Client Computer System
137
Client Computer System
143
Server Computer System
147
Files
09563145.100300

FIG. 1

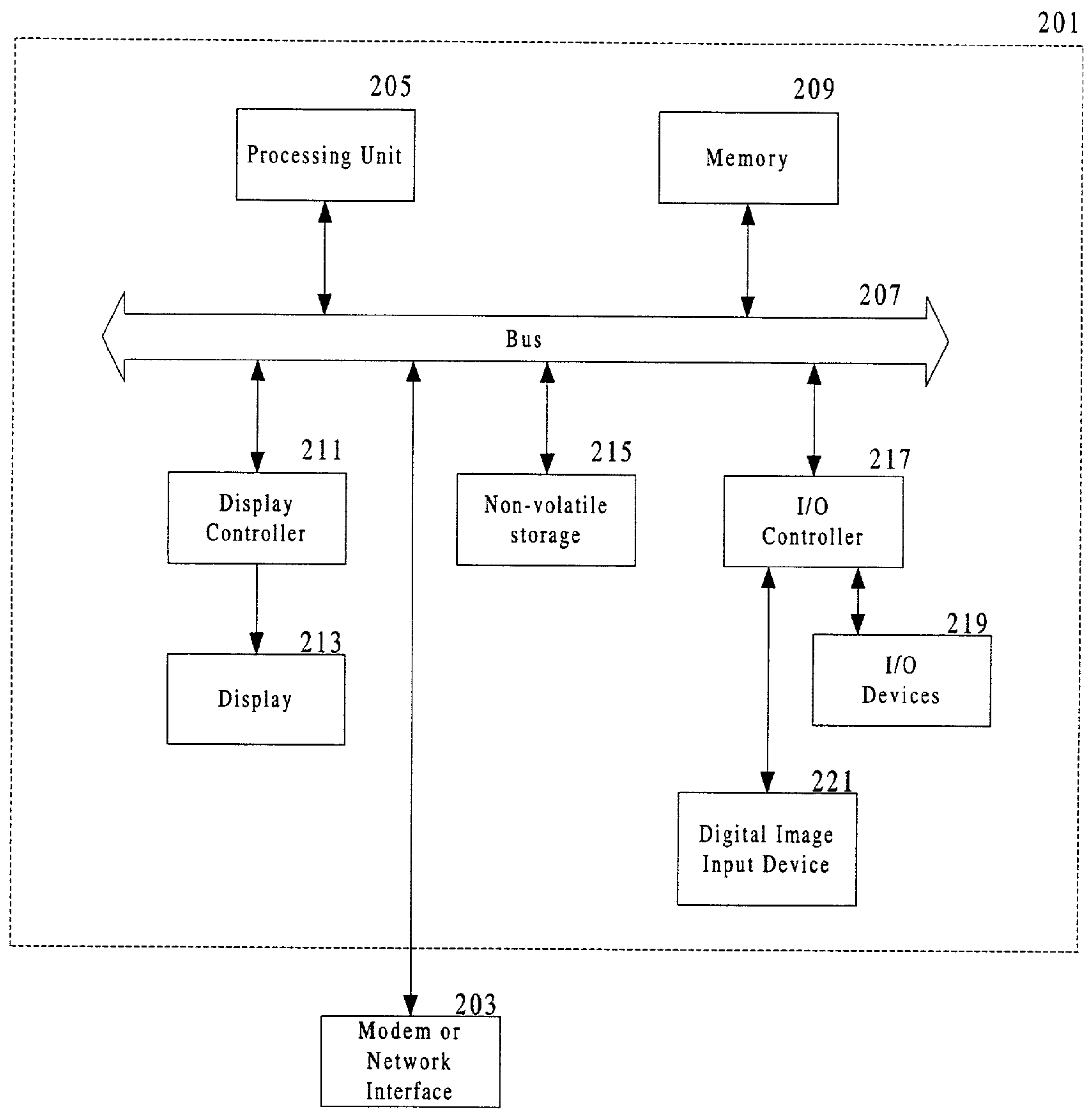
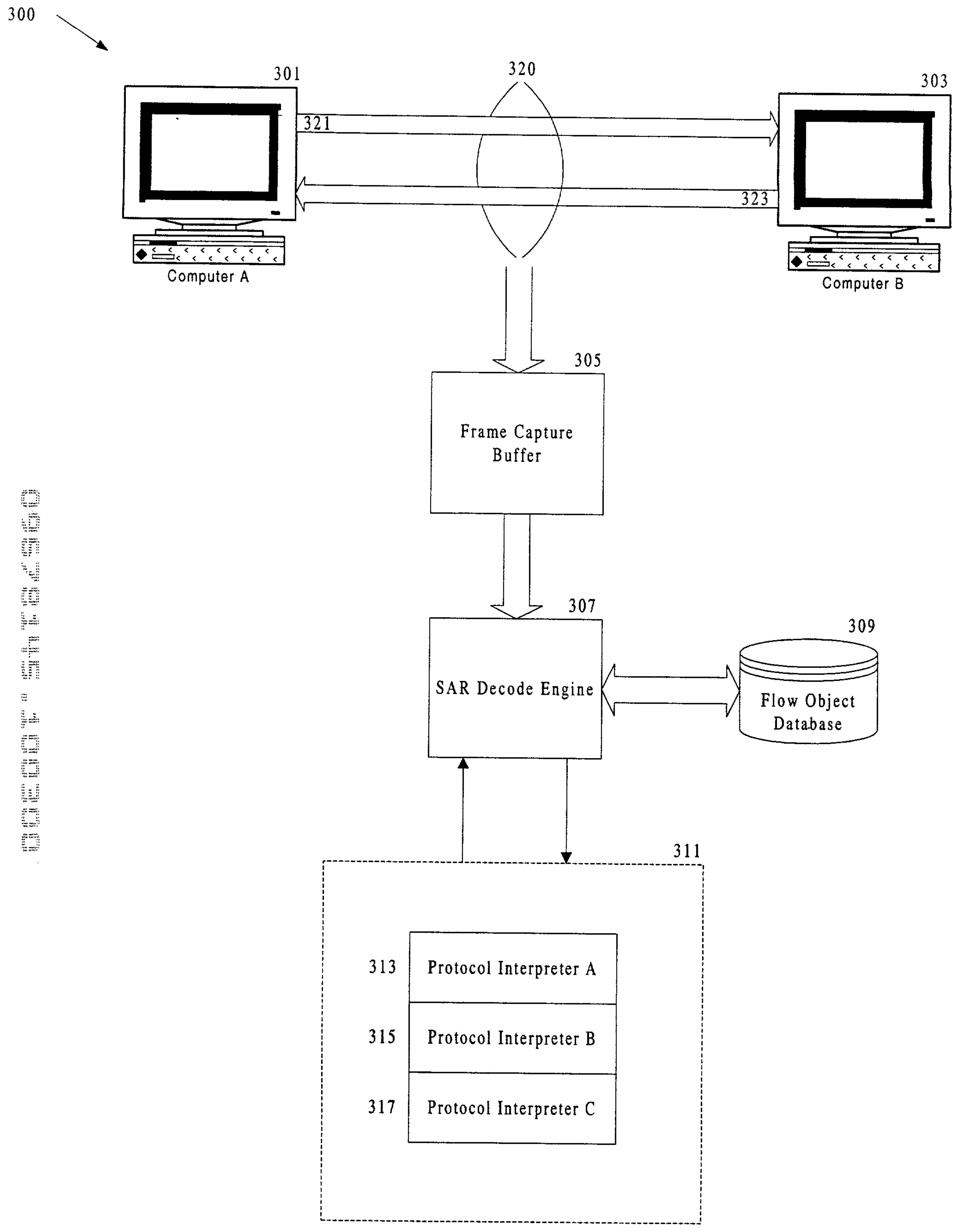

201
205
Processing Unit
209
Memory
207
Bus
211
Display Controller
213
Display
215
Non-volatile storage
217
I/O Controller
219
I/O Devices
221
Digital Image Input Device
203
Modem or Network Interface

FIG. 2



300
301
320
303
321
323
Computer A
Computer B
305
Frame Capture Buffer
307
SAR Decode Engine
309
Flow Object Database
311
313
Protocol Interpreter A
315
Protocol Interpreter B
317
Protocol Interpreter C

FIG. 3

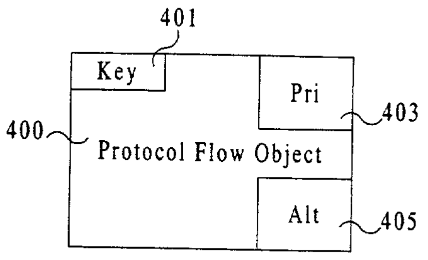

401
Key
Pri
403
400
Protocol Flow Object
Alt
405

FIG. 4A

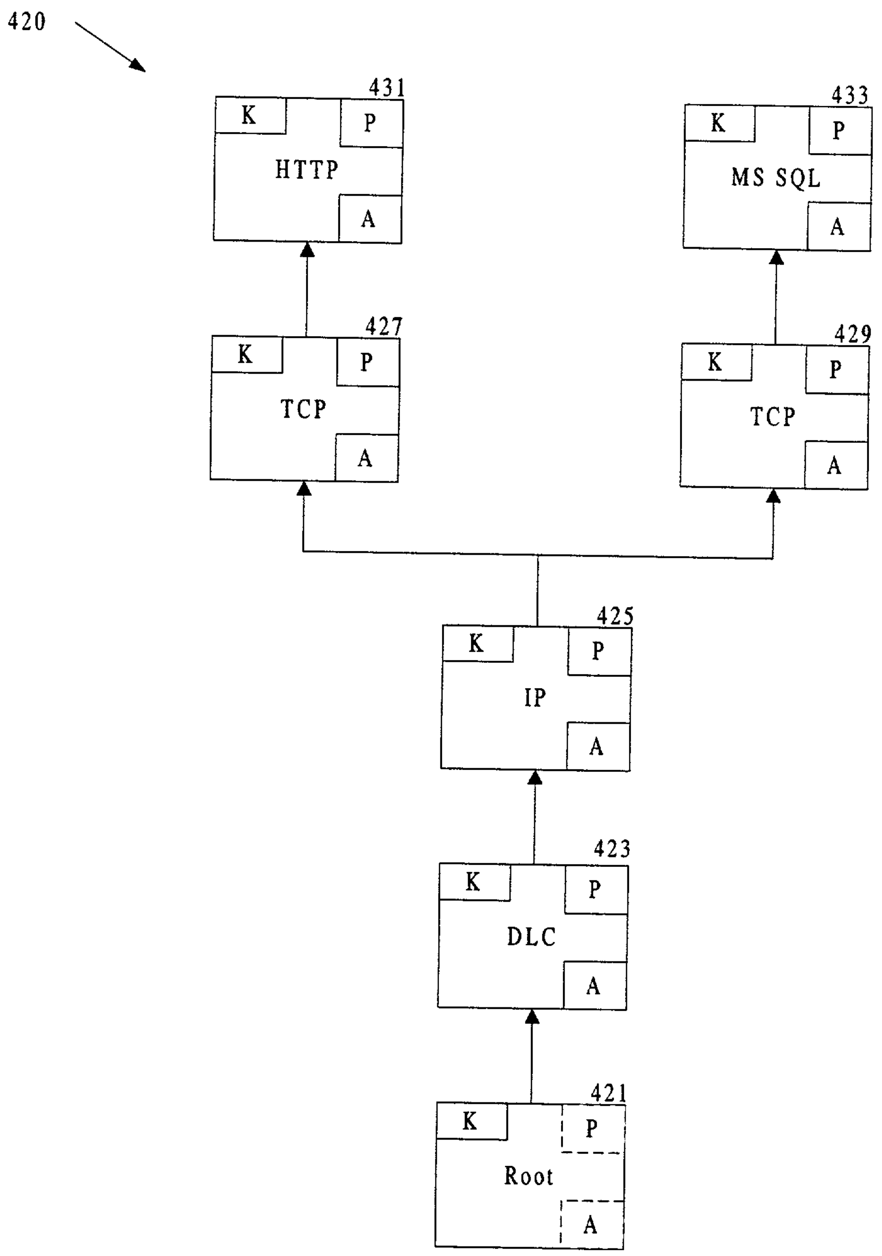
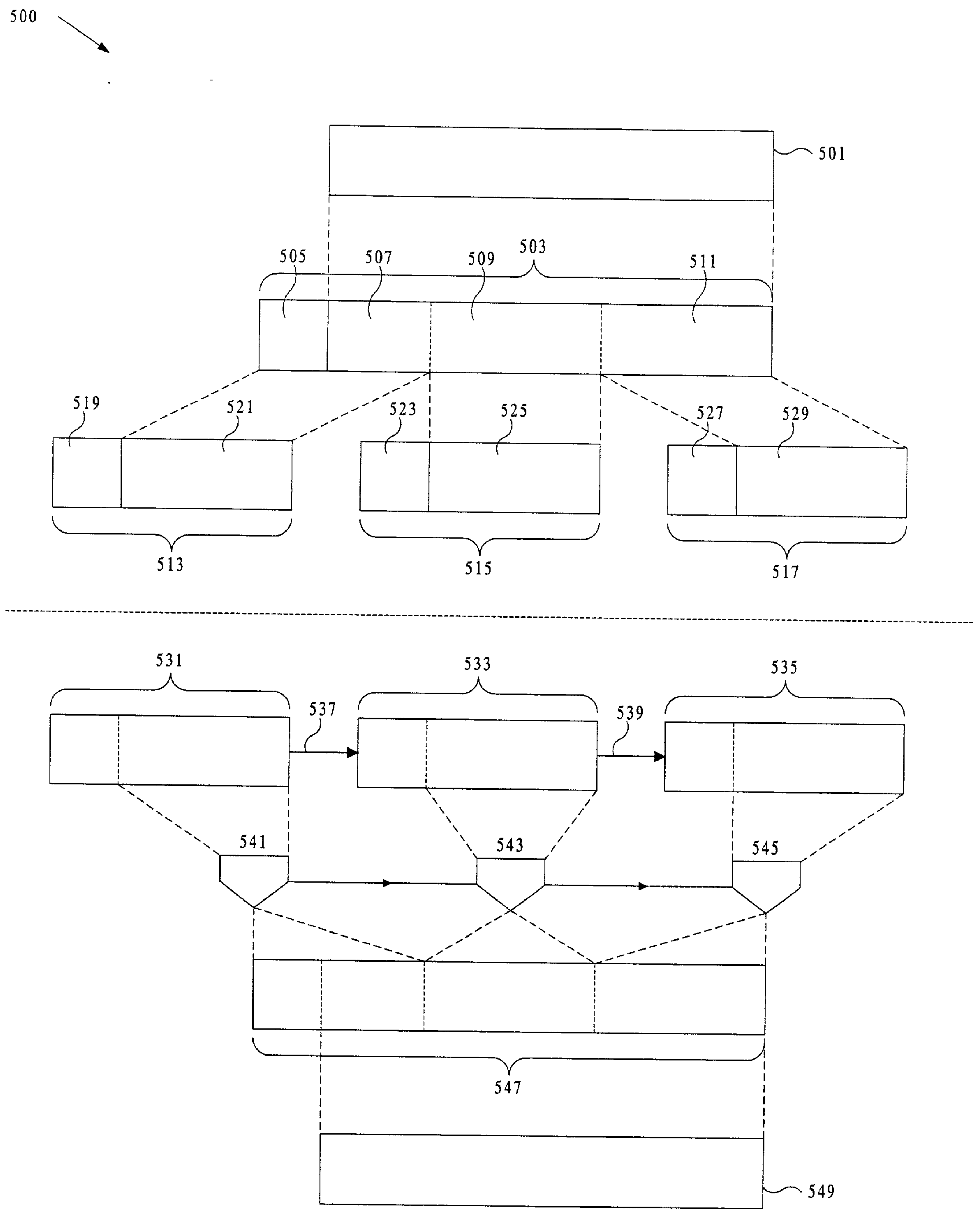

420
431
K
P
HTTP
A
433
K
P
MS SQL
A
427
K
P
TCP
A
429
K
P
TCP
A
425
K
P
IP
A
423
K
P
DLC
A
421
K
P
Root
A

FIG. 4B



500
501
503
505
507
509
511
513
515
517
519
521
523
525
527
529
531
533
535
537
539
541
543
545
547
549

FIG. 5A

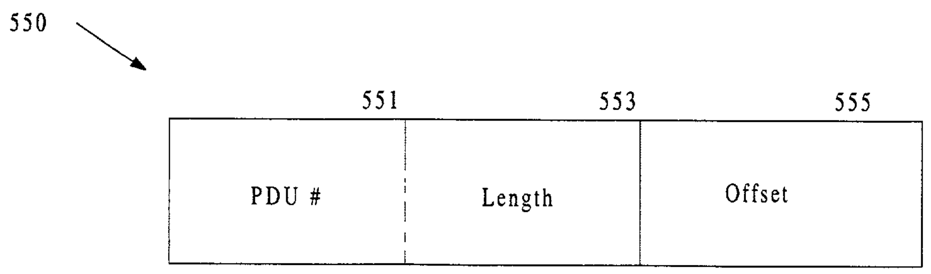
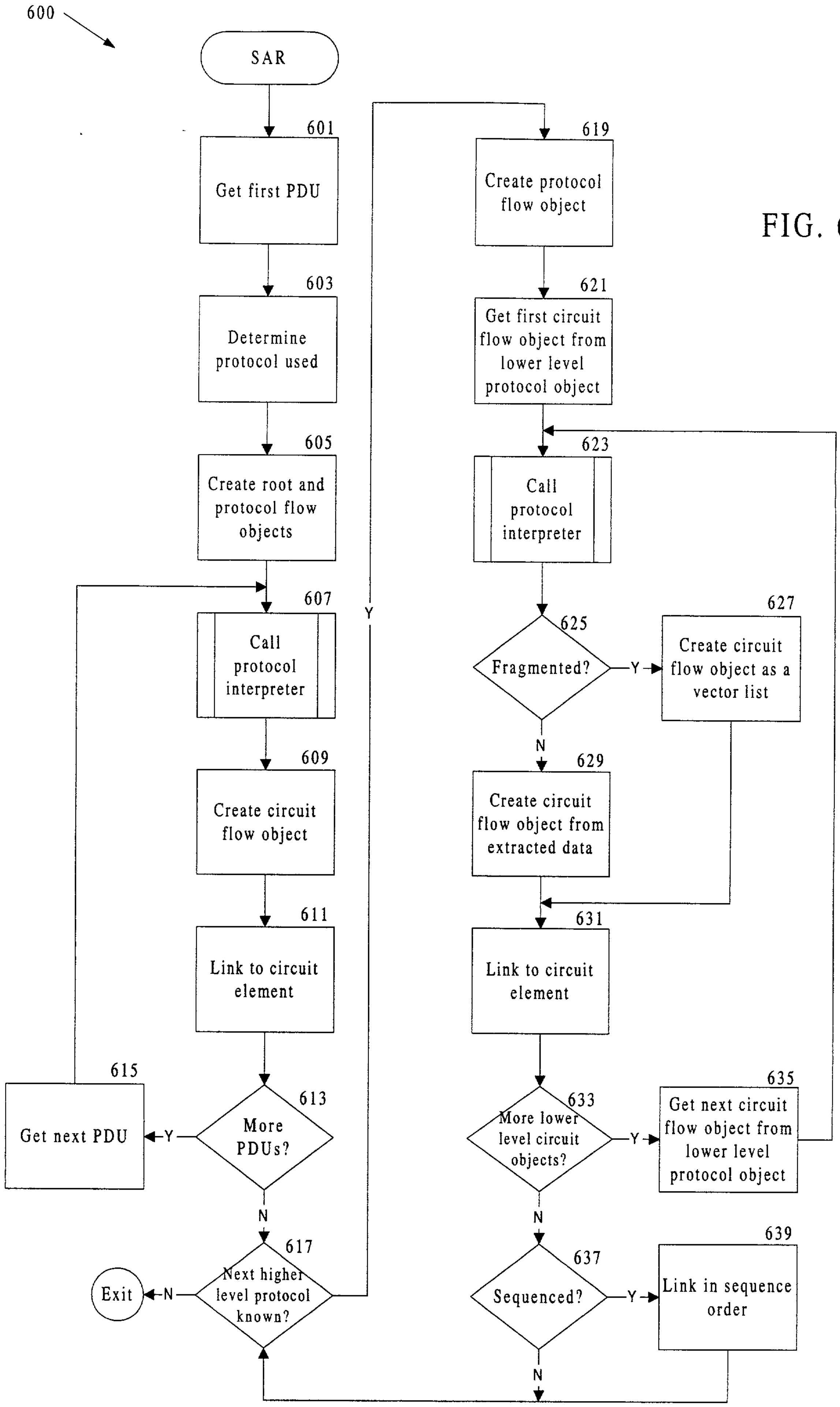


FIG. 5B



600
SAR
601
Get first PDU
603
Determine protocol used
605
Create root and protocol flow objects
607
Call protocol interpreter
609
Create circuit flow object
611
Link to circuit element
613
More PDUs?
Y
615
Get next PDU
N
617
Next higher level protocol known?
N
Exit
Y
619
Create protocol flow object
621
Get first circuit flow object from lower level protocol object
623
Call protocol interpreter
625
Fragmented?
Y
627
Create circuit flow object as a vector list
N
629
Create circuit flow object from extracted data
631
Link to circuit element
633
More lower level circuit objects?
Y
635
Get next circuit flow object from lower level protocol object
N
637
Sequenced?
Y
639
Link in sequence order
N

FIG. 6A

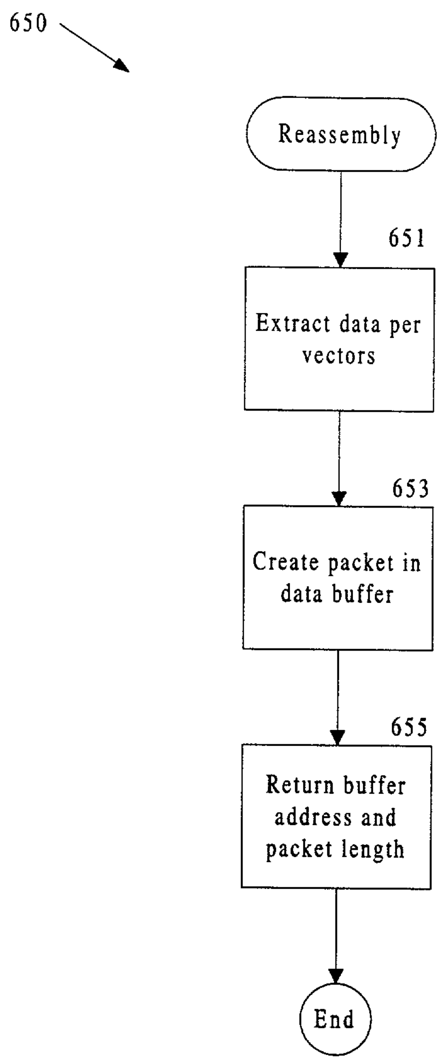
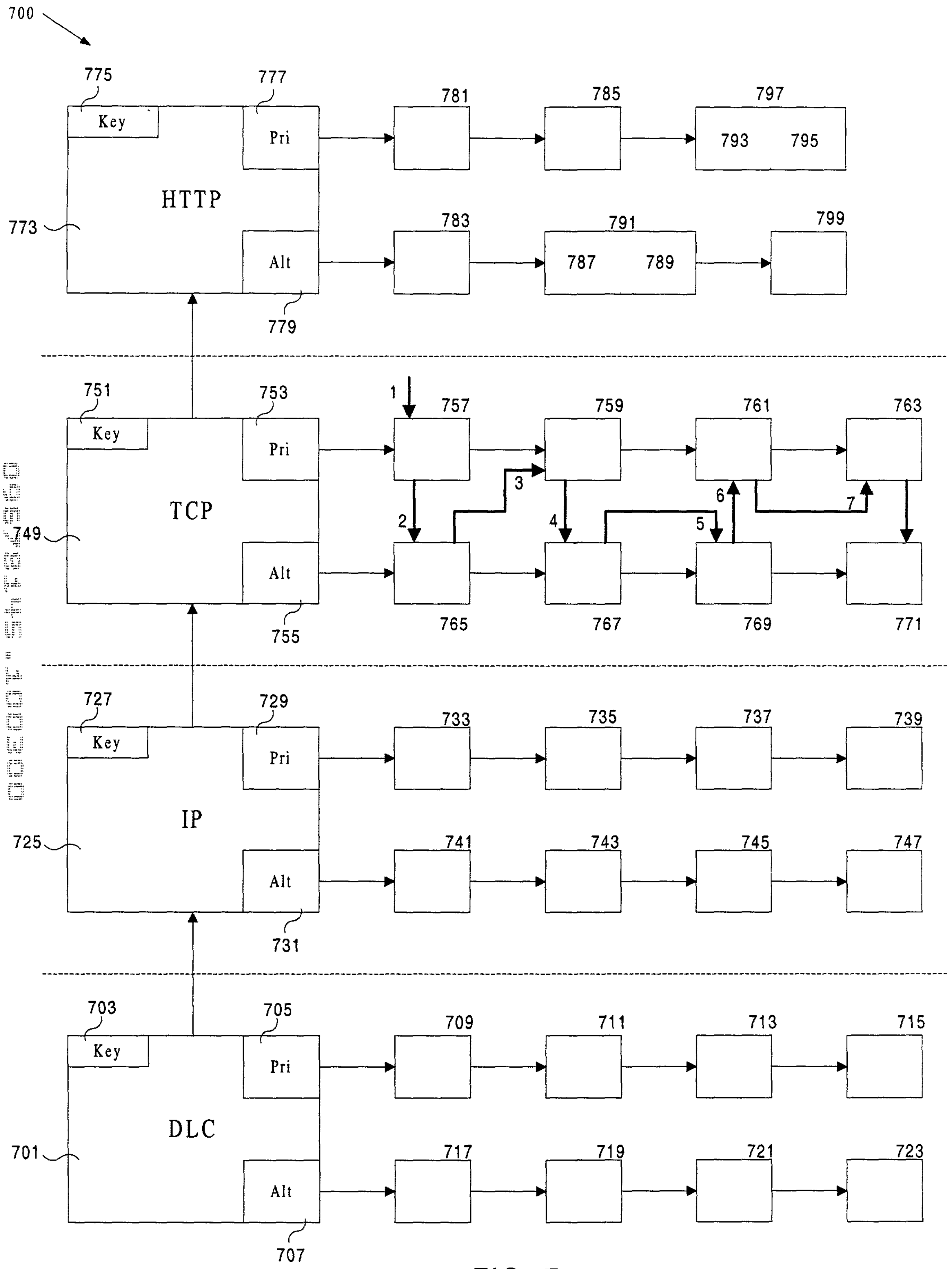

650
Reassembly
651
Extract data per vectors
653
Create packet in data buffer
655
Return buffer address and packet length
End

FIG. 6B



700
775
Key
777
Pri
HTTP
773
Alt
779
781
785
797
793
795
783
791
787
789
799
751
Key
753
Pri
TCP
749
Alt
755
1
757
759
761
763
2
3
4
5
6
7
765
767
769
771
727
Key
729
Pri
IP
725
Alt
731
733
735
737
739
741
743
745
747
703
Key
705
Pri
DLC
701
Alt
707
709
711
713
715
717
719
721
723

FIG. 7

Attorney's Docket No.: 002114.P012 Patent

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled
MULTI-LAYER PROTOCOL REASSEMBLY THAT OPERATES INDEPENDENTLY OF UNDERLYING PROTOCOLS, AND RESULTING VECTOR LIST CORRESPONDING THERETO

the specification of which

X is attached hereto.
___ was filed on ______ as
United States Application Number ______
or PCT International Application Number ______
and was amended on ______.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

(Application Number)	Filing Date
(Application Number)	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Number)	Filing Date	(Status -- patented, pending, abandoned)
(Application Number)	Filing Date	(Status -- patented, pending, abandoned)

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Sheryl Sue Holloway, BLAKELY, SOKOLOFF, TAYLOR &
(Name of Attorney or Agent)
ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct telephone calls to Sheryl Sue Holloway, (408) 720-8300.
(Name of Attorney or Agent)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Stuart John Macdonald

Inventor's Signature [illegible] Date 9/29/2000

Residence Littleton, Colorado (City, State) Citizenship United States (Country)

Post Office Address 49 Dawn Heath Circle
Littleton, Colorado 80127

Full Name of Second/Joint Inventor Jerome Norman Freedman

Inventor's Signature ____________ Date ____________

Residence Greenbrae, California (City, State) Citizenship United States (Country)

Post Office Address 373 North Almenar Drive
Greenbrae, California 94904

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Stuart John Macdonald

Inventor's Signature ____________________ Date ____________

Residence Littleton, Colorado (City, State) Citizenship United States (Country)

Post Office Address 49 Dawn Heath Circle
Littleton, Colorado 80127

Full Name of Second/Joint Inventor Jerome Norman Freedman, Ph.D.

Inventor's Signature JNFreedman Date ____________

Residence Greenbrae, California (City, State) Citizenship United States (Country)

Post Office Address 373 North Almenar Drive
Greenbrae, California 94904

09574585.140300

APPENDIX A

William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Lisa N. Benado, Reg. No. 39,995; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; R. Alan Burnett, Reg. No. 46,149; Gregory D. Caldwell, Reg. No. 39,926; Andrew C. Chen, Reg. No. 43,544; Thomas M. Coester, Reg. No. 39,637; Donna Jo Coningsby, Reg. No. 41,684; Florin Corie, Reg. No. 46,244; Dennis M. deGuzman, Reg. No. 41,702; Stephen M. De Klerk, Reg. No. P46,503; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Robert Andrew Diehl, Reg. No. 40,992; Sanjeet Dutta, Reg. No. P46,145; Matthew C. Fagan, Reg. No. 37,542; Tarek N. Fahmi, Reg. No. 41,402; George Fountain, Reg. No. 37,374; Paramita Ghosh, Reg. No. 42,806; James Y. Go, Reg. No. 40,621; James A. Henry, Reg. No. 41,064; Libby N. Ho, Reg. No. P46,774; Willmore F. Holbrow III, Reg. No. P41,845; Sheryl Sue Holloway, Reg. No. 37,850; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; William W. Kidd, Reg. No. 31,772; Sang Hui Kim, Reg. No. 40,450; Walter T. Kim, Reg. No. 42,731; Eric T. King, Reg. No. 44,188; Erica W. Kuo, Reg. No. 42,775; George Brian Leavell, Reg. No. 45,436; Kurt P. Leyendecker, Reg. No. 42,799; Gordon R. Lindeen III, Reg. No. 33,192; Jan Carol Little, Reg. No. 41,181; Joseph Lutz, Reg. No. 43,765; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Clive D. Menezes, Reg. No. 45,493; Chun M. Ng, Reg. No. 36,878; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Daniel E. Ovanezian, Reg. No. 41,236; Kenneth B. Paley, Reg. No. 38,989; Marina Portnova, Reg. No. 45,750; William F. Ryann, Reg. 44,313; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; John F. Travis, Reg. No. 43,203; Joseph A. Twarowski, Reg. No. 42,191; Tom Van Zandt, Reg. No. 43,219; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Mark L. Watson, Reg. No. P46,322; Thomas C. Webster, Reg. No. P46,154; Steven D. Yates, Reg. No. 42,242; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and Firasat Ali, Reg. No. 45,715; and Justin M. Dillon, Reg. No. 42,486; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and James R. Thein, Reg. No. 31,710, my patent attorney with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

APPENDIX B

Title 37, Code of Federal Regulations, Section 1.56
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclosure all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.